Question - 1

SCORE: 50 points

Fun with Anagrams

Data Structures | Strings | Problem Solving | Easy

Two strings are anagrams if they are permutations of each other. For example, "aaagmnrs" is an anagram of "anagrams". Given an array of strings, remove each string that is an anagram of an earlier string, then return the remaining array in sorted order.

For example, given the strings *s = ['code', 'doce', 'ecod', 'framer', 'frame'],* the strings *'doce'* and *'ecod'* are both anagrams of *'code'* so they are removed from the list. The words *'frame'* and *'framer'* are not anagrams due to the extra *'r'* in *'framer'*, so they remain. The final list of strings in alphabetical order is *['code', 'frame', 'framer'].*

Function Description

Complete the function *funWithAnagrams* in the editor below. It must return a list of strings in alphabetical order, ascending.

funWithAnagrams has the following parameters:

s[s[0],...s[n-1]]: an array of strings

Constraints

- *1 ≤ n ≤ 1000*
- *1 ≤ |s[i]| ≤ 1000*
- *Each string s[i] is made up of characters in the range ascii[a-z]*

▼ Input Format For Custom Testing

The first line contains an integer, *n*, that denotes the number of elements in *s*.

Each line *i* of the *n* subsequent lines (where *0 ≤ i < n*) contains a string that describes *s[i]*.

▼ Sample Case 0

Sample Input For Custom Testing

```
4
code
aaagmnrs
anagrams
doce
```

Sample Output

```
aaagmnrs
code
```

Explanation

aaagmnrs and *anagrams* are anagrams, *code* and *doce* are anagrams. After sorting *aaagmnrs* comes first.

Help

▼ Sample Case 1

Sample Input For Custom Testing

```
4
poke
pkoe
okpe
ekop
```

Sample Output

```
poke
```

Explanation

All of the strings are anagrams of each other. Only the first occurrence is left.

Question - 2

SCORE: **50 points**

Shortest Substring

Algorithms | Strings | Data Structures | Problem Solving | Easy | Sets | HashMaps

Given a string comprised of lowercase letters in the range *ascii[a-z]*, determine the length of the smallest substring that contains all of the letters present in the string.

For example, given the string *s = dabbcabcd*, the list of all characters in the string is *[a, b, c, d]*. Two of the substrings that contain all letters are *dabbc* and *abcd*. The shortest substring containing all the letters is *4* characters long, *abcd*.

Function Description

Complete the function *shortestSubstring* in the editor below. The function must return the length of the shortest substring that contains all of the characters within *s*.

shortestSubstring has the following parameter:

s: a string

Constraints

- $1 \leq size\ of\ s \leq 10^5$
- $s[i] \in ascii['a'-'z']$

▼ Input Format For Custom Testing

The first line contains a string, *s*.

▼ Sample Case 0

Sample Input For Custom Testing

```
bab
```

Sample Output

```
2
```

Explanation

"ba" is a substring that contains all the characters in *s*.

▼ Sample Case 1

Sample Input For Custom Testing

```
bcaacbc
```

Sample Output

```
3
```

Explanation

"bca" is a substring that contains all the characters in *s*.